Amendments to the Drawings are presented on page 64 of this paper, and replacement drawings are submitted for approval.

Remarks begin on page 65 of this paper.

An **Appendix** including replacement drawing figures is attached following page 66 of this paper.

IN THE SPECIFICATION:

Page 1, substitute paragraph [0001] and [0002] as follows:

Field of the Invention

[0001] The invention ~~concerns~~relates to a method and a system for generating a global simulation model of an architecture. More particularly, the invention ~~concerns~~relates to a configuration method and a system called a "Configurator" for implementing the method.

Background

[0002] With the increasing complexity of hardware systems, it is necessary, in order to verify systems or integrated circuits under design, to be able to handle configurations that increasingly include models written in a hardware description language, for example of the HDL type (such as ~~VDHL~~VHDL or Verilog, the most widely used), and in a high level languages of the HLL type (such as C or C^{++}); these languages describe, on the one hand, the elements constituting the hardware, and on the other hand, the models constituting the simulation environment.

Please substitute paragraphs [0008] through [0024] as follows:

[0008] ~~The~~One object of the present invention is to limit one or more of these drawbacks.

Summary of the Invention

[0009] To this end, the invention primarily ~~concerns~~relates to a method of automatic generation, by means of a data processing system associated with a program called a Configurator for creating a global simulation model of an architecture comprising models of integrated circuits under development that can constitute, with the help of the automatic Configurator, a machine or a part of a machine, and environment simulation models that make it possible to test and verify the circuit under development, a configuration definition file for components of the architecture, these components constituting fixed functional blocks for describing the functionalities of integrated circuits or parts of integrated circuits, the components being chosen by the user from a library of various component types and a library of environment components, in order to create the global model of the architecture corresponding to the functional specification defined in the configuration definition file and conforming to the specification of the architecture of the global model specified by an architecture description file, ~~a method characterized in that it includes the following steps~~ comprising :

- ~~the~~ reading ~~of~~ the architecture description file of the global model and the storage, in a component and connection rule table, in a connection coherency rule table, and in a source file formatting table, of information related to all of the possible configurations, with each component obtaining ~~an~~a name that unambiguously identifies its position in the architecture, and a type from among several types (Active Components, Monitoring and Verification Blocks, Intermediate Blocks, System Blocks and Global Blocks)~~,~~;

- ~~the instantiation of~~instantiating the components specified in the configuration definition file by the user-developer using a list of the components present, designated by their names and types and including parameters or invoking procedures, the configuration definition file comprising a file from which ~~to select the~~ components ~~and~~are selected together with their types and optional additional indications concerning the type of interface and the server involved in the configuration to be generated by the Configurator, and ~~the storage of~~storing the corresponding information in ~~an~~the instance connection table~~,~~;

- ~~the topological connection of~~topologically connecting the instances and

~~the storage of~~storing the corresponding information in ~~an~~the instance connection table~~;~~;

- ~~the physical connection of~~physically connecting the interface signals, at the level of each instance of the components, by applying regular expressions, stored in the component and connection rule table, based on the names of the signals constituting a wiring table, and

- ~~the use of~~using the instance connection table, the wiring table and the formatting table to automatically generate HDL-type and HLL-type source files of the global simulation model corresponding to the configuration specified by the configuration definition file.

[0010] According to another characteristic of the method, the Configurator system transmits to the HLL-type parts of each component information on:

- the name of the component~~;~~,
- the type of the instance, and
- the HDL path, i.e. the hierarchical name of the component in the description of the model.

[0011] According to another characteristic of the method, the configuration definition file also includes a keyword indicating the name or number of the server in which a component is instantiated, when the method is used in a multi-server system.

[0012] According to another characteristic of the method, in the case of a multi-server utilization, the Configurator system executes the following steps:

- the division of the Configuration into several (HDL-type and HLL-type) parts, sorting the HDL-type components and the HLL-type objects according to the servers to which they belong,
- the generation of the HDL-type peripheral components used for sending and receiving signals between the parts of the configuration,
- the duplication of the Global Blocks by the Configurator system and the instantiation of the Global Blocks duplicated in each server, and

- the generation of HLL-type parts that serve as a communication medium between the servers.

[0013] According to another characteristic of the method, the automatic connection between the components by the Configurator system includes several phases:

- a high-level topological phase for selecting the components and their respective positions,
- a wiring phase for creating the actual connection between the components, this phase generating as a result a wiring table that associates the signals connected to one other with the unique name of the wire that connects them, and
- a phase for generating HDL-type and HLL-type source files.

[0014] According to another characteristic of the method, the wiring phase ~~is~~ performed by the Configurator system ~~in the following three steps~~comprises:

a. connecting the Global Blocks and the System Blocks ~~are first connected~~ to all of the components~~,~~;

b. ~~next come the connections of~~connecting, or "wiring", the signals between the ~~other components,~~;

c. ~~after the wiring,~~making an additional pass ~~makes it possible~~ to connect the remaining unconnected signals of each component to predetermined signals in order to produce a given stable state; ~~the Configurator system then generates~~and

d. generating, via the Configurator system, partial configurations comprising a subset of the architecture.

[0015] According to another characteristic of the method, the predetermined signals are the signals of the System Block corresponding to the component.

[0016] According to another characteristic, the description file of the architecture of the global model includes the simulation models of ~~the.~~ ~~Global~~theGlobal Blocks and the System Blocks, these two types of components being connected to one another and handling environment signals.

[0017] According to another characteristic, the System Blocks are connected to the other components and supply them with the system signals that are specific to them.

[0018] According to another characteristic of the method, the data processing system performs a conformity check of the connections, comparing the connection table of the real instances between blocks to the connection coherency rule table.

[0019] According to another characteristic of the method, the data processing system compares the physical connections between the components to the connection coherency rule table, in order to detect any incompatibilities between the ends of the connections between the components, and in such a case, it specifies, and adds into the instance connection table, an adapter component inserted into the connection in question.

[0020] According to another characteristic, the configuration definition file includes information, specified by an attribute, concerning the utilization of adapter components with the instances of the active Components, whose connections are compared by the data processing system to the instance connection table in order to detect any incompatibilities between the ends of the connections between the components, and in such a case where incompatibility is detected it specifies, and adds into the instance connection table, another adapter component inserted into the connection in question. According to another characteristic of the method, the data processing system selects certain connections between the components of the connection coherency rule table and specifies, and adds into the instance connection table, additional connections constituting

branches leading to respective additional models, which represent tools for monitoring the connections.

[0021] According to another characteristic of the method, ~~in~~during the source file generation phase, the Configurator system generates the source files in HDL language and in HLL language, based on the content of the component and connection rule table, the connection coherency rule table, the source file formatting table, the instance connection table and the wiring table.

[0022] According to another characteristic of the method, the data processing system executes an operation through the Configurator system for each configuration variant, in order to obtain several simulation models corresponding to the same functional specification, but written in a description comprising various mixtures of languages of different levels (HDL, HLL).

[0023] According to another characteristic of the method, the data processing system generates the functional specification of the global simulation model in a computer format compatible with a high-level programming language, (HLL) and in a format compatible with a hardware description language (HDL).

[0024] According to another characteristic of the method, the configuration definition file comprises, for each component, at least one part in HDL-type language, ~~said part in HDL-type language providing~~in order to provide an interface with other models.

Please substitute paragraphs [0037] through [0072] as follows:

[0037] According to another characteristic of the system, the Configurator system uses:

- an HLL base class for uniquely identifying each object instantiated and for polling the configuration~~,~~;

- a means for generating and automatically instantiating System Blocks~~,~~;

- a means for generating tables associating the signals connected together under the unique name of the connecting wire~~,~~; and

- a means for using a formatting table to generate the source files in HDL and HLL.

[0038] According to another characteristic of the system, ~~the~~an operator functionally specifies the configuration in the highest level language as much as possible, and completes the functional specification with the components in the lowest level language.

[0039] According to another characteristic of the system, the following entries in the hash define the component Type (for example DUT (HDL model), XACTOR (transactor), MONITOR, VERIFIER or other types), and ~~corresponds~~correlate each Component Type to a hash, in turn composed of the following entries:

- a first entry *ReqModule* contains the name of the HDL module of the component and the name of the corresponding source file,
- a second entry *Connect* is the definition of the method for selecting the signals that are part of a Port, this description being composed of a set of entries indexed by the name of the Port; the ~~configurator~~Configurator associates each Port name with a table of regular expressions and a pointer to a signal connection procedure that controls the application of these expressions to the names of the signals of the interface of the component.

[0040] According to another characteristic of the system, the generic structure of the active Components includes a Block containing the HDL description and a Block in HLL that provides the access paths to the HDL resources~~,~~ and, if necessary, a description of the block in HLL; the set of signals of the HDL block constitute the interface of the containing Block, formed by Ports, which are arbitrary logical selections of the signals of an interface, and by interface adapters, which are the software parts that handle, in each Port, the two-way communication between the parts in HLL and those in HDL, the interface adapters being chosen by the Configurator system.

[0041] According to another characteristic of the system, the Ports are specified in the form of regular expressions, which serve both to select the subsets of signals to be connected and to define the connection rules.

[0042] According to another characteristic of the system, the Configurator system generates so-called Transfer Components, which are inserted on each side of the cutoff to the servers, these components simply being wires for the inputs and registers for the outputs.

[0043] The elementary model components, which are shared among the Composite Model Components or the Global Blocks belonging to the various servers, are automatically duplicated by the Configurator system and instantiated in each server.

Brief Description of the Drawings

[0044] The invention will be better understood with the help of the following description of a preferred embodiment of the method of the invention, in reference to the attached drawings, in which:

- Fig. 1 represents, in highly schematic form, an exemplary architecture of the global simulation model for an integrated circuit (ASIC) under development, called BRIDGE (12);
- Figs. 2A through 2D are diagrams illustrating the various components of the Configurator system and the steps for implementing these components in the method of the invention;
- Figs. 3a through 3c represent various stages in the modeling of a circuit using a mixed HDL (VERILOG ~~or VDHL~~orVHDL) type and the HLL (C^{++}) type mode;~~,~~[[]]
- Fig. 4 represents the generic structure of an elementary Model;
- Fig. 5 represents the generic structure of a Composite Model;
- Fig. 6 describes the correspondence between the tables of the description of the configurator system in Figs. 2a through 2d and the tables for implementing the method of the invention;
- Figs-7a through 7k schematically represent the various models of the components with the necessary variants of their interfaces and description levels for the configurations related to the architecture in the example of Fig. 1; and
- Figs. 8a through 8e represent the successive configurations of the architecture that results in the creation of a given model, the final model of which

corresponds to that of Fig. 8e, for which all of the HDL-type models of the circuit under design are available, this final model corresponding to the configuration; Fig. 8f represents the distribution of the simulation configuration of the model of Fig. 8a, for example into two machines.

Description of the Preferred Embodiments

[0045] The invention concerns a system called a "Configurator," and the configuration method implemented by the system.

[0046] A global simulation model is typically composed of one or more models of integrated circuits under test (DUTs) surrounded by models that create a ~~testing~~test and verification environment. These models create complex stimuli and receive complex responses from the model being tested. These components can be transactors (XACTORS) – models that generally have ~~a program~~an application programming interface (API) that permits control by tests outside the model, these tests generally being written in a high level language (HLL).

[0047] The verification environment can also contain components called Monitoring Blocks (MONITOR) and components called Verification Blocks (VERIFIER). These components are not directly involved in the exchange of signals between the other components of the global simulation model, but are used to observe and interpret them. The Monitoring Blocks (MONITOR) serve as analysis aids for the tests; they have program interfaces (APIs) for reporting events observed in the signals of the global model. The Verification Blocks (VERIFIER) are components that have a reference specification for the operation of the model being tested, and, by observing the signals of the global simulation model, they are capable of verifying the proper operation of the model.

[0048] Fig. 1 represents an exemplary architecture of an integrated circuit system under development for which it is necessary, in a first step, to debug the global simulation model corresponding to Fig. 8c, chosen in order to illustrate the greatest number of mechanisms implemented by the ~~configurator~~Configurator. It should be clear that the steps in Figs. 8a, 8b, could be executed first, depending on the availability of the models and the objectives of the verification. The final model desired is the one that corresponds to Fig. 8e. The architecture is constituted by a processor (CPU) communicating through a bridge (BRIDGE) with a system memory (MEMORY) and input/outputs (I/O). The model produced by the "Configurator" system of the invention is constituted by a processor component CPU of the XACTOR type, connected by an

interface of the "fbus_p" type to an intermediate block (fbus_xchg) 101 having a different type of interface. Another intermediate block (fbus_xchg) (102), in Figs. 8b and 8c, connects the first intermediate block (101) to a bridge-type component (BRIDGE) of the DUT_CORE type that communicates with a model, written primarily in an HDL-type language, of a memory (MEMORY) of the DUT type, with a model written primarily in an HDL-type language of an input/output (I/O) of the DUT type, and with a system block (SYS_BRIDGE).

[0049] The "Configurator" system of the invention handles the connection of software simulation elements called components for the purpose of simulating hardware circuits.

[0050] There are at least 5 classes of components:

- ~~the~~ "Active Components" (see below);
- ~~the~~ "Monitoring and Verification Blocks" (see below);
- ~~the~~ "Intermediate Blocks" (see below);
- ~~the~~ "System Blocks" (see below); and
- ~~the~~ "Global Blocks:[[]] (see below and 1 of A2).

[0051] Each type of component can have several variants of embodiment of the same element, differentiated by the description level (see below) or by the signals in the interfaces (see below). Each type of ~~Component~~component can be described in several levels of detail (functional, behavioral, gates, etc.), in an HDL-type language like VERILOG or VHDL, or in a high level language (HLL) like C or C^{++}, complemented by an HDL-type interface.

[0052] Several description levels for describing similar functionalities can coexist and can have HDL-type interfaces that are similar but not necessarily identical. Certain descriptions can have more functionalities, and the HDL-type interfaces can contain completely different sets of signals.

[0053] The components are connected based on a predetermined schema that is considered to be fixed for each execution of the Configurator system. These connections are defined by the architecture of the global simulation model and specified by the architecture description file (FDARCH) (see ~~Annexes~~Appendices A1-A5).

[0054] Each instance of a component in this schema obtains a name or label that unambiguously identifies the position of the component and its type.

[0055] The definition file of the configuration (FCONF) provides a synthetic description of the Configuration variant to be generated by the Configurator system. Only

the main components (Active Components, Monitoring Blocks and Verification Blocks) are mentioned in it, along with the types of models desired. The other components (Global Blocks, System Blocks and Intermediate Blocks) are instantiated automatically by the Configurator system.

[0056] Among the various types of components, the "Active Components" constitute the subset of the objects explicitly designated in the configuration definition file (FCONF) that exchange stimuli with their environment by transmitting and receiving.

[0057] Among the various types of components, the "Monitoring and Verification Blocks" constitute the subset of the objects explicitly designated in the configuration definition file (FCONF) that merely receive information from the environment. They are used for purposes of observation and Verification (MONITOR, VERIFIER). The operation granularity of these models is the event, which reports changes in control signal values and arbitrary data exchanges.

[0058] All the other Components constitute so-called implicit components, which are managed and instantiated automatically by the Configurator system, as a function of the parameters of the configuration definition file (FCONF) (explicit component type, interface type, etc.).

[0059] The components can be connected to each other directly or via external adaptation components called "Intermediate Blocks," specially defined and declared for this purpose. They are often used, as will be seen below, during successive development phases to complete the missing parts of the design.

[0060] The Configurator system may thus insert one or more intermediate blocks to connect two active components.

[0061] The components called "System Blocks" are associated with the other components in order to supply them with the environment signals that are specific to them. These components encapsulate, at the level of each interface, all of the system signals that do not participate in the connection between the other components. The "System Blocks" themselves are connected to the "Global Blocks," and manage all of the environment signals, i.e., the clock signals and general control signals (clock, reset, powergood, diagnostic), which may be transformed in order to adapt them to the needs of the corresponding active block (polarity change, delay, etc.), as well as the specific signals that are different for each particular instance of the active component in question, for example, the signals that encode the module number or the operating mode of the component, etc. The latter are managed by parameters provided by the Configurator

system to the instances of the models of the components generated. If, for a given Component, the System Block is not necessary (which is indicated by the description tables of the configuration), it will be connected directly to the Global Blocks (see 1 of A2).

[0062] The configuration definition file (FCONF) can contain additional information specifying intermediate blocks to be used in association with the instances of the active components. Thus, ~~the~~a user can influence the choice of the intermediate component, or eliminate the ambiguity if there are several possible choices. The connections thus obtained are compared to the connection coherency rule table (TRCOH) in order to detect any incompatibilities between the ends of the connections between the components, and in such a case to choose, and add into the instance connection table (TCINST), another adapter component (Intermediate Block), inserted into the connection in question.

[0063] The Configurator system is based on a generic representation, as described in Figs 3a through 3c, common to all of the components declared in the architecture description file (FDARCH) of the global simulation model and comprising two parts, of the HDL type (for example VERILOG or VHDL) and HLL type (for example C^{++}).

[0064] In the case of a component described entirely in an HDL-type language (Fig. 3a), the HLL-type part is reduced to one instance, which makes it possible to indicate its presence in the Configuration during the simulation and supplies the paths for access to the HDL-type resources of the component.

[0065] For components described in an HLL-type language (Fig. 3c), it is the HDL-type part that is reduced to a strict minimum and is limited to the description of the interface registers and signals.

[0066] All of the intermediate levels between these two extremes are possible, and are naturally used in the context of processes for developing ASIC circuits.

[0067] Typically, during development, one begins with an HLL-type model with a minimal HDL-type interface, then moves on to increasingly complete HDL-type models written by the designers, and ends up with models on the logic gate level that are automatically synthesized from HDL-type models.

[0068] Mixed models are often used because they allow for a precise and efficient simulation by dedicating the high-level language (HLL) to complex ~~modelings~~modeling for which the HLL-type language offers a compact and quickly

executed solution. Nevertheless, even for models written primarily in an HLL-type language, the HDL-type part can be non-trivial because of performance losses due to changes in the simulation context between the HDL-type and HLL-type parts, respectively. A typical example is an interface whose signals change, even without any real data transfer activity. In this case, it is preferable to process the signals in the HDL-type part of the model, and to switch to the HLL-type part when the data are present in the interface.

[0069] The interface of a component is the set of all the signals present on its periphery. For components described in an HDL-type language only, this corresponds to the external interface of the definition of this component in an HDL-type language (for example VERILOG or VHDL). For components defined in a mix of HLL- and HDL-type languages, the interface of a component is the sum of the signals of all the HDL-type models used on the periphery of this component.

[0070] Fig. 4 describes the generic structure of the elementary models that constitute the majority of the components. This structure typically, though not exclusively, applies, in the case of the ASIC circuit under development, to interface adapters, intermediate blocks, and system blocks.

[0071] It includes a containing block, marked C, containing the HDL-type description marked A, and the HLL Block marked B that provide the paths for access to the HDL-type resources and, if necessary, a description of the block in an HLL-type language. The set of signals of the HDL-type block constitutes the interface of the block C. For purposes of connecting between the blocks, we will use the concept of Ports (see below), which are arbitrary logical selections of the signals of an interface. It is possible for a signal to belong to several Ports at once.

[0072] The Configurator system is capable of handling so-called "composite" models comprising a part in an HLL-type language and including other models of components called "interface adapters." Interface adapters are mixed HDL/HLL-type models having ~~a program~~an application programming interface (API) in an HLL-type language through which they communicate with the composite model. Composite models are particularly useful for simulation environment components (for example MONITOR, VERIFIER, XACTORS) wherein the model must adapt to signals present in the different variants of the models used in a configuration.

Please substitute paragraphs [0078] through [0092] as follows:

[0078] The Configurator system is designed to optimize the utilization of ~~the~~ Probes, while keeping their numbers to a minimum. Since the description of the Components establishes the notion of equivalency between certain Components, the Configurator system first tries to share the port of an "Active Component." If this proves impossible, it instantiates a suitable Probe component that can be connected to the HDL-type interface from a list provided in the description of the Monitoring or Verification Component.

[0079] The section below explains the definitions related to the connections between Components. The concepts of interfaces and ports are used to support the description of the connections between the components.

[0080] ~~Let us recall that the~~A port is an arbitrary selection of signals of the HDL-type interface of a component, performed using regular expressions based on the names of these signals. A given signal can belong to one or more ports. The definition of each port is ~~constituted by~~comprised of pairs of regular expressions and corresponding substitute expressions. These expressions are successively applied to the name of each signal of the interface, and in case of an appropriate "match," the substitute expression is applied, and the name thus obtained moves on to the next substitution operation. If the final substitution gives a final result that is identical for two signals, ~~the latter~~they will be connected to one another. The ~~configurator~~Configurator generates a unique name for the connection and places the appropriate information in the wiring table (TCAB). The method described makes it possible to express complex connection rules between two components. For example, it is possible to connect signals with different names, or to create rules for connecting the input signals with the output signals.

[0081] The creation of this method for definition by the interfaces and the ports makes it possible to separate the declarations of HDL-type models and the specifications of connections for the Configurator system. The latter combines these two sources of information in order to generate the connections. In the majority of cases, modifying declarations in the HDL-type parts, which happens quite frequently during development, does not entail any modifications in the regular expressions that describe the wiring.

[0082] In the exemplary embodiment of the invention discussed below, only the point-to-point connections are described. The "bus"-type connections are easily modeled using a component with several outputs incorporating the bus.

[0083] The method of automatic connection between the "components" by the

Configurator system in order to create the global simulation model will be described below. This method comprises the following steps:

- ~~The~~ reading ~~of~~ the architecture description file (FDARCH) of the global model and ~~the storage~~storing, in a component and connection rule table, in a connection coherency rule table, and in a source file formatting table, ~~of~~the information related to all of the possible configurations, with each component obtaining a name that unambiguously identifies its position in the architecture, and a type from among several types (Active Components, Monitoring and Verification Blocks, Intermediate Blocks, System Blocks and Global Blocks)~~.~~;
- ~~The instantiation of~~instantiating the components specified in the configuration definition file by the user-developer using a list of the components present, designated by their names and types and including parameters or invoking procedures, the configuration definition file comprising a file from which to select the components and their types and optional additional indications concerning the type of interface and the server involved in the configuration to be generated by the Configurator, and ~~the storage of~~storing the corresponding information in an instance connection table~~.~~;
- ~~The topological connection of~~topologically connecting the instances based on the schema defined by the component and connection rule table (TCRC).
- ~~The~~ reading and ~~storage of~~storing the HDL-type sources of the models instantiated, in order to extract the names and types of the interface signals from them~~.~~;
- ~~The~~ wiring ~~of~~ the interface signals, at the level of each instance of the components by applying regular expressions, stored in the component and

connection rule table (TCRC), based on the names of the signals constituting the wiring table (TCAB)~~.~~; and

- ~~The use of~~using the instance connection table (TCINST), the wiring table (TCAB) and ~~the~~a formatting table (TFMT) to automatically generate the HDL-type (MGHDL) and HLL-type (MGHLL) source files of the global simulation model corresponding to the configuration specified by the configuration definition file (FCONF).

[0084] The topological phase proceeds in the following way:

- The explicit ~~Components~~components specified in the configuration definition file (FCONF) are ~~first~~ instantiated by the Configurator system and placed in the instance connection table (TCINST). This table contains the description of each instance (label, type), accompanied by a list of the components with which it is connected~~.~~;
- ~~Next,~~ the Intermediate Blocks, explicitly specified in the ports of the "Active Components" in the configuration definition file, are instantiated and added to the instance connection table (TCINST)~~.~~;
- ~~The~~the connections between the active components instantiated are compared with the rules contained in the connection coherency table (TRCOH) in order to detect any incompatibilities between the ends of the connections between the components, and in that case to choose, and add to the instance connection table (TCINST), an adapter component (Intermediate Block) to be inserted into the connection in question~~.~~;
- ~~Next,~~ the components of the Monitoring Block and Verification Block type are instantiated. The Configurator system analyzes the connections of the composite model components and searches for sharable interface adapters at

the level of the common ports (see the models C_Port$_i$ in Fig. 3). If there is no suitable component with which to share the connection by observing the required signals, an interface adapter component with the properties specified by the description will be instantiated by the ~~configurator~~Configurator. The instance connection table (TCINST) is updated~~.~~; and

- ~~Lastly,~~ the "system block" components are instantiated and placed in the instance connection table (TCINST) for the components that need them.

[0085] In the Wiring phase, the Configurator system connects the interface signals at the level of each port by applying regular expressions based on the names of the signals as described in the definition of the port. These expressions are applied to the names of the signals extracted from the HDL-type descriptions of the components, so any changes in the HDL-type interfaces from one version of a model to another do not directly affect the specifications of the connections between the blocks. This phase results in the generation of the wiring table (TCAB), which associates the signals connected to one another with the unique name of the wire that connects them. A certain number of verifications are then possible, for example, on the sourceless connections, ~~or~~ the connection of several outputs, or other connections.

[0086] The various steps in the wiring phase are the following:

- ~~The~~the Global Components and System Blocks are ~~first~~ wired to all of the components~~.~~;
- ~~Next,~~ the signals between the other components are wired~~.~~; and
- ~~After the wiring,~~ an additional pass ~~makes~~is performed to determine if it is possible to connect the signals of a component not connected in previous wiring phases to the corresponding signals of the system block, this pass is specially provided for forcing known and stable values that inhibit the unused

ports. The signals in question of the system blocks carry a special marking (for example a special prefix in the name) so as not to be connected during the previous wiring phases.

[0087] In ~~a~~the source file generation phase, the Configurator system generates the HDL-type and HLL-type source files based on the content of the instance connection table (TCINST), the wiring table (TCAB) and the formatting table (TFMT), in order to automatically generate the HDL-type (MGHDL) and HLL-type (MGHLL) source files of the global simulation model corresponding to the configuration specified by the configuration definition file (FCONF).

[0088] The HDL-type source file contains a description of the HDL-type part (for example in VERILOG) of the global simulation model corresponding to the configuration definition file (FCONF). In particular, it contains the instantiation of all of the HDL-type models of the components mentioned in the configuration file, the intermediate blocks, System Blocks and Global Blocks, as well as all of the wires connecting these instances.

[0089] The HLL-type source file contains the program that corresponds to the instantiation of all the components having part of their models in an HLL-type language. In the case of object-oriented HLL languages, the code contains the constructors of HLL class objects with the corresponding parameters specified in the description of each component. A formatting table (TFMT) specific to each design makes it possible to generate the appropriate HLL-type code.

[0090] ~~We will now describe the~~The architecture description file (FDARCH) for a specific implementation of the ~~configurator~~Configurator system (PROGCONF) is described below in the PERL language~~,~~. PERL was chosen because of ~~the~~its direct manipulation of ~~the~~ regular expressions and ~~the~~ associative tables on several levels.

[0091] The architecture, represented in Fig. 1, is constituted by a processor (CPU) communicating through a bridge (BRIDGE) with a system memory (MEMORY) and input/outputs (I/O).

[0092] In order to facilitate its writing and manipulation, the file FDARCH represented has been divided into several parts, presented in ~~Annexes~~Appendices A1 through A5. This file defines the architecture of the global simulation model corresponding to the generic diagram represented in Fig. 1. The HDL-type language generated is VERILOG and the high level language (HLL) generated is C^{++}.

Please substitute paragraph [0095] as follows:

[0095] The description of the rules for interconnecting the components (TCRC) contains global parameters common to all of the component types. It can exist in the form of at least one table. In the exemplary embodiment of Fig. 6, it is shown in the form of seven associated (hash) tables, three of which have the entries (%InstNameModuleMap, %SysConnectMap, %SysSpecMap), the names designating all of the possible models for the same component:

- The table "%InstNameModuleMap" (see 2 of A1), which appears in the file patent_config_defs.pm for describing the explicit ~~Components~~components. This table represents the rules for generating connections between the signals. It has the entries Connect and SelfConnect.
- The table "%SysConnectMap," which appears in the file patent_Sysconfig_defs.pm for the System Blocks.
- The table "%SysSpecMap," (see 4 of A2), which appears in the file patent_Sysconfig_defs.pm for the Intermediate Blocks.

Please substitute paragraphs [0101] through [0124] as follows:

[0101] Other parameters are optional and can be added to the description of the tables, but these parameters can only be used by the code provided with the description; among them are the following:

- NumAdd, which is an auxiliary parameter serving as a modifier for the parameters extracted by ExtrExpr, used by the procedure GenDest (see below)~~.~~, and
- BaseldExpr, which is a regular expression that makes it possible to find the active component corresponding to an intermediate or system component in case of ambiguity.

[0102] The following hash entries define the Type of the component, for example DUT (HDL-type model), XACTOR (transactor) or other types.

[0103] Each Component Type corresponds to a hash, in turn composed of the following entries:

- ~~The entry~~ *ReqModule* , which contains the name of the HDL-type (VERILOG) module of the component and the name of the corresponding source file~~,~~; and
- ~~The entry~~ "Connect", which is the definition of the method for selecting the signals that are part of a Port. This description is composed of a set of entries indexed by the name of the Port. Each Port name is associated with a table of regular expressions and a pointer to a procedure for connecting the signals, which controls the application of these expression to the names of the signals of the interface of the component.

[0104] The Configurator has several preprogrammed procedures, but others can be added and referenced. The preprogrammed procedures accept the following modifiers:

= ~~-~~E (exclusive – the signal will be used only once)~~.~~;

- ~~=~~T (branch - the signal can be used by several destinations~~).~~; and
- ~~=~~A special value "filter_out_wire_name" assigned to a signal or a group of signals makes it possible to eliminate any connection to the corresponding signals.

[0105] The entry "SelfConnect" -- similar to the entry "Connect" -- contains the definitions of the connections of the Port signals that can be connected between two instances of the same component. This definition facilitates the connection of one-way output ~~?~~/ input signals in the particular case of a head-to-tail connection.

[0106] The entry "Port" provides the definition of all of the ports. It is a hash that associates the name of each port with a table that includes the following elements:

- ~~The~~the logical name of the HDL-type component~~.~~;
- ~~The~~the name of the HDL-type module or a choice of modules~~.~~; and
- ~~The~~the number of the port (useful for generating VERILOG and C^{++} code).

[0107] The entry "GenDest" (see 1 of A3) is a pointer to the procedure that, for a component designated by a label and type, generates, for each port, the label of the component to which it is connected. The ~~referenced procedure~~user must imperatively ~~be specified by~~specify the ~~user~~referenced procedure; this procedure is also used to control the coherency of the requested configuration.

[0108] The entry "SysConn" is a hash that associates each HDL-type module contained in the model with a pointer to the procedure for selecting and naming the System Blocks. The ~~referenced procedure~~user must imperatively ~~be specified by~~specify the ~~user~~referenced procedure.

[0109] The entry "GenHDLInstParam" is a pointer to the procedure that generates the additional parameters required by the HDL-type simulator for the

code generated, which instantiates the HDL-type part of the component.

[0110] The entry "CfgCpp" defines the parameters of the constructor of the object for the C^{++} identification code of the Configuration, generated automatically, which serves as a filter for the selection of the Components by the Configurator system.

[0111] The first parameter is the name of the class of the C^{++} object; it is a predefined name associated with that of the HLD-type model. ~~It~~The first parameter is followed by ~~the~~ tables that correspond to groups of parameters typically associated with each port of the component. The keyword "Own" indicates an elementary model component.

[0112] Next comes the name of the HDL-type component referenced in the Port part and the type identifier of the component, for example DUT, model of the circuit under design, XACTOR, or transactor.

[0113] For the composite model blocks, the parameter part is richer.

[0114] It contains the keyword "Src" for the active components or "Mon" for the Verification and Monitoring components.

[0115] Next come the parameters of the interface adapter component, comprising:

- the name of the HDL-type component~~.~~; and
- the identifier of the component type and its class followed by the name of the corresponding port.

[0116] The latter parameters are specific to each port of the composite block.

[0117] In the C^{++} code generated for a composite model, the parameters transmitted to the constructor of a class correspond to the pointers to the instances of the

interface adapters.

[0118] All of these parameters serve to guide the generator of the C++ code, which combines them with the rules contained in the hash table "%classCppProtoMap":

- ~~The~~the entry "ProbeConnect" associates each port of a Monitor or Verifier component with the C++ class that is acceptable as an interface adapter. Thus, the Configurator is capable of optimizing the code by using the same interface adapter for several components (composite model)~~.~~; and
- ~~The~~the entry "SysWrap" is a hash that, for each HDL-type part of a component, defines the regular expression to be used for setting the remaining unconnected signals to a known state after the wiring between the components.

[0119] The following algorithm is used for connecting two components ~~is the following~~:

- ~~The~~the Configurator system ~~first tries~~attempts direct connection~~.~~,
- ~~If this~~if direct connection proves impossible based on the first hash table %HwfConnectivityMap, the modules specified in %HwifAlternateMap are retrieved in order to be placed in the component. The optional modules are indicated by the character ">" at the beginning of their name in the specification of a component.
- ~~In~~in the end, if no module works, the Configurator system returns to the hash %HwfConnectivityMap to choose the intermediate blocks to be placed between the components~~.~~; and
- ~~If~~if no suitable block is found, an error is signaled.

[0120] The user can influence the Configurator system's choice by explicitly specifying in the configuration file the intermediate blocks to be connected to a component.

[0121] The table "%HwfConnectivityMap" allows for both the Verification of connectivity and the specification of intermediate Blocks to be inserted in order to connect two Components whose parts are not directly connectable.

[0122] In order for two blocks to be able to be connected to each other, it is necessary for %HwfConnectivityMap to contain entries that describe the following properties of this connection:

- ~~Each~~each component is associated with the component to which it can be connected;
- ~~A~~a direct connection is indicated by the value "Direct";
- ~~For~~for connections requiring an intermediate Block, the name of this block is indicated; and
- ~~For~~for certain blocks, the connectivity can be specified differently for each port of the block. This serves to eliminate the ambiguity in the connections of a block. A typical utilization involves the insertion of two intermediate blocks connected head-to-tail between two explicit components.

[0123] It is emphasized that the connectivity in this case is expressed globally, without specifying the names of signals to be connected.

[0124] The Components described in an HLL-type language can have several possible implementations of the HDL-type modules A_Port, at the level of each port i (see Fig. 3). The hash table "%HwifAlternateMap" specifies the choice that is possible for each HDL-type module of a component. The ~~configurator~~Configurator system uses this information in order to insert the intermediate blocks only when strictly necessary.

Please substitute paragraph [0130] as follows:

[0130] The hash %Activity_TypeMap (see 1 of A1) associates the name of a component with its type, specified by ~~a keyword~~one of the following keywords:

- "actor" for the ~~active components.~~Active Components;
- "spectator" for the Monitoring blocks~~.~~;
- "checker" for the Verification blocks~~.~~; or
- "helper" for the System blocks or the Intermediate blocks in case of an indirect connection.

Please substitute paragraphs [0137] through [0139] as follows:

[0137] The hash %classCppProtoMap describes the generation of the constructors of C++ class objects. Each entry corresponds to the name of a C++ class. Each description is a hash composed of the following four entries:

- ~~The entry~~ "Prea", which corresponds to the generation of the first part of the constructor (class, name of the instance, first parameters)~~.~~;
- ~~The entry~~ "Iter", which corresponds to the iterative part of the parameters of the constructor. This rule is applied for each element contained in the entry CfgCpp of the description of the component~~.~~;
- ~~The entry~~ "Post", which corresponds to the last part of the constructor (last parameters, closing parentheses)~~.~~; and
- ~~The entry~~ "Idle", which is a rule that substitutes for the rule "Iter" for nonexistent connections (partial configurations). Typically, a null pointer or null string is generated.

[0138] Each entry is a character string containing predetermined special selectors, replaced by the values calculated by the Configurator system. These selectors have the general form #<Src¦Dst¦Act><Item#>, where "Src," "Dst" and "Act" indicate the instance to be used, the latter being transmitted as a parameter to a constructor of an HLL object.

- "Src" indicates the current instance~~.~~;
- "Dst" indicates the instance connected to the other end of the port~~.~~, and
- "Act~~.~~" indicates the composite instance corresponding to the Active Component containing the observation port.

[0139] <Item> is a keyword indicating the selection of one of the following:

- "Inst" for the instance name cpp~~.~~;
- "Iid" for the component label corresponding to Inst~~.~~;
- "Ict" for the component type (DUT, VERIFIER, XACTOR, Monitor, etc.)~~.~~; or
- "Port" for the name of the port.

Please substitute paragraph [0147] as follows:

[0147] The following HDL-type description of the Interfaces has been reduced to only what is necessary to illustrate the capabilities and the operation of the ~~configurator~~Configurator. In a real situation, the behavioral descriptions are included, as in the case of the model of the clock (the module CLOCK).

Please substitute paragraphs [0169] through [0175] as follows:

[0169] The connectivity rules that make it possible to generate the connectivity tables of any configuration for this example are described in PERL language in ~~Annexes~~Appendices A1 through A5.

[0170] ~~Annex~~Appendix A1 describes, in the case of the example chosen, the topological rules linked to the active components to be used by the Configurator system.

[0171] ~~Annex~~Appendix A2 describes the topological rules linked to the system Blocks and the Intermediate blocks to be used by the Configurator system.

[0172] ~~Annex~~Appendix A3 describes the procedures for connecting the Components to be used by the Configurator system.

[0173] ~~Annex~~Appendix A4 describes the code for generating the file in HDL-type language (Verilog_top.v) for connecting the components to be used by the Configurator system.

[0174] ~~Annex~~Appendix A5 describes the code for generating C^{++} Classes.

[0175] ~~Annexes~~Appendices A6 and A7 describe the result files generated by the Configurator system, respectively in a low level HDL-type language (VERILOG) and in a high level language (C^{++}), in order to create the model of the configuration defined by the third configuration corresponding to Fig. 8c. The ~~configurator~~Configurator can thus, with the user-developer, create its own model in accordance with the configuration variant chosen.

Please substitute paragraphs [0181] through [0189] as follows:

[0181] A memory 53 contains the instance connection table (TCINST) representing the instances of the components and their mutual interconnections required by the Configuration defined in the configuration definition file (FCONF) and conforming to the rules contained in the tables TCRC and TRCOH.

[0182] The memory 53 also contains the wiring table (TCAB) representing the physical connections (wires) between the HDL-type parts of the components instantiated from the HDL-type source file library (BFSHDL).

[0183] This results in the files MGHDL and MGHLL, which respectively represent the HDL-type (VERILOG or VHDL) and HLL-type (C^{++}) source files of the global simulation model generated by the Configurator system, a model that allows the co-simulation of an ASIC circuit in its verification environment.

[0184] The implementation of the method of the invention will now be explained.

[0185] The method for automatically generating, by means of the computer or data processing system 40, a simulation model of a configuration of models of given components, mutually linked by interworking connections so as to create a global simulation model MGHDL/MGHLL of an ASIC circuit in its verification environment, ~~includes the following steps~~comprises (see Fig. 2a):

- an operator, possibly aided or even replaced by a computer, ~~generates~~generating the architecture description file (FDARCH) describing the rules for interconnection between the various components, the models being capable of being used to model each of these components and the formatting rules in order to generate the source files of the resulting model~~;~~;
- an operator, possibly aided or even replaced by a computer, ~~generates~~generating the source file library BFSHDL from HDL-type parts of

models of respective components capable of being used in a configuration in conformity with the content of the file FDARCH;.

- ~~the~~an operator ~~generates~~generating the desired configuration definition file (FCONF), which identifies the components and the desired models (see Fig. 2b)~~.~~;
- ~~The~~ data processing system ~~reads~~reading the architecture description file (FDARCH) and generates and places in memory the tables TCRC, TRCOH and TFMT from FDARCH (see Fig. 2b)~~.~~;
- ~~The~~the various tables TCRC, TRCOH and TFMT having been made accessible to the data processing system 40 (see Fig. ~~2c),~~2c);
- the data processing system 40 ~~reads~~reading the desired configuration definition file (FCONF) in order to identify the components and their required models (see Fig. ~~2c);~~2c);
- ~~it reads~~the data processing system 40 reading the component and connection rule table in order to instantiate the required components and place them in the instance connection table TCINST (see Fig. 2c)~~.~~;
- ~~It reads~~the data processing system 40 reading the connection coherency rule table (TRCOH) in order to verify the physical connectivity between models and, if necessary, inserts intermediate components~~.~~;
- ~~It reads~~the data processing system 40 reading, in the library BFSHDL, the source files corresponding to the HDL-type parts of the component models instantiated in the table TCINST in order to generate the table TCAB of physical connections (wires) between the component models (see Fig. 2c)~~.~~; and

- ~~It~~ the data processing system 40 applies the formatting rules from the table TFMT to the contents of the tables TCINST and TCAB, and it generates the final source files MGHDL/MGHLL of the global simulation model corresponding to the configuration specified by the configuration definition file (FCONF) (see Fig. 2d).

[0186] The input file FCONC corresponds to a description of the theoretical diagram of the type in Fig. 1, but without the models, which are added during the generation of the intermediate tables TCINST, TCAB describing the configuration and connections, in order to provide a global simulation model that is actually operational and that, moreover, allows the observation of particular signals.

[0187] The various tables above can be pre-stored in the ~~computer~~data processing system 40 or can even be stored in an external memory that one connects to the ~~computer~~data processing system 40, possibly through a data transmission network, in order to make them accessible.

[0188] The ALU 42 has all of the basic data required by the table tools TCRC, TRCOH, TFMT in order for the program PROGCONF associated with the Configurator system to control, in the ALU 42, the generation of the final files MGHDL/MGHLL (see Fig. 2d).

[0189] The intermediate table TCAB describing the physical connections generated from an HDL-type description, is, in this example, subjected to a verification in order to detect any obvious errors, such as the existence of unconnected inputs or outputs or even outputs connected in parallel, when there should be conventional outputs of the so-called "totem-pole" type, as opposed to those of the open collector or tri-state type.

Please substitute the title of the Annex at page 44 as follows:

~~ANNEX~~APPENDIX: Description of the Configurator Code

Please substitute the title of the Annex at pages 50, 55, 57, 59, 63, and 67 as follows:

~~ANNEX~~APPENDIX